Purpose

This guide is a supplement to SANS FOR572: Advanced Network Forensics: Threat Hunting, Analysis, and Incident Response. It covers some of what we consider the more useful Linux shell primitives and core utilities. These can be exceedingly helpful when automating analysis processes, generating output that can be copied and pasted into a report or spreadsheet document, or supporting quick-turn responses when a full tool kit is not available.

Remember: If you can make it happen in a shell over a lag-ridden SSH connection, there is a better chance of being the lethal forensicator when it really matters!

How To Use This Document

Linux has been around since 1991, and its *NIX parents since 1969. This handout cannot begin to scratch the surface of the great and powerful things you can do with nothing more than a shell prompt and some moxie. Use this document as a "memory jog" for some of the capabilities of the more commonly used tools in this course and in the forensic workflow in general.

Dig into the details of each tool's features through its manual pages (aka "man pages") and other online and offline references. We think you will find the shell to be as powerful as the GUI, and in some cases a far superior alternative – especially for scalability and automation.

No Packets, No Party

tcpdump – Dump network traffic

```
$ sudo tcpdump -n -s 0 -i eth0 '<BPF filter>'
$ tcpdump -n -r input.pcap -w output.pcap '<BPF filter>'
```

- **-n** Prevent DNS lookups on IP addresses. Use twice to also prevent port-to-service lookups
- **-r** Read from pcap file instead of the network
- **-w** Write packet data to a file
- **-D** Enumerate network interfaces
- **-i** Specify the network interface on which to capture
- **-s** Number of bytes per packet to capture
- **-C** Number of megabytes to save in a capture file before starting a new file
- **-G** Number of seconds to save in each capture file (requires time format in output filename)
- **-W** Used with the -C or -G options, limit the number of rotated files (see man page for detailed usage)
- **-x** Display packet contents in hex

tcpdump *requires root privileges to capture network traffic promiscuously. User-level permissions are sufficient for manipulating existing capture files.*

See the ***pcap-filter*** *man page for information on building BPFs to control captured traffic.*

Index That pcap File!

nfpcapd – Generate **nfdump**-compatible NetFlow records from **pcap** file

```
$ nfpcapd -r in.pcap -l ./netflow/ -S 1 -z
```

- **-r** pcap file to read
- **-l** Output directory
- **-s** Output directory format (0=flat, 1=yr/mo/day; see **nfcapd** man page for more)
- **-z** Compress output flows

Fundamental Usage: Pretty Print

nfdump – Process NetFlow data from files on disk

```
$ nfdump -R ./ -O tstart -o extended
```

- **-R** Recursively read data from the specified directory
- **-r** Read data from a single nfcapd file
- **-a** Aggregate by src+dst IP, src+dst port, protocol
- **-A** Specify custom aggregation
- **-t** Time window, in "**YYYY/MM/DD.hh:mm:ss**" format (See man page for additional details)
- **-s** Generate "TopN" statistics
- **-O** Specify output ordering
- **-o** Specify output format (**line**, **long**, **extended**, or custom). Custom formatting uses "**fmt:<format string>**" syntax, where "**<format string>**" defines values displayed (see man page for full list).

%ts	Start time	**%te**	End time
%td	Duration	**%pr**	Protocol
%sa	Source address	**%da**	Destination address
%sap	Source IP:port	**%dap**	Destination IP:port
%sp	Source port	**%dp**	Destination port
%sas	Source ASN	**%das**	Destination ASN
%pkt	Packet count	**%byt**	Byte count
%fl	Flow count	**%flg**	TCP flags
%bps	Bits per second	**%pps**	Packets per second
%bpp	Bytes per packet		

What's in a Name?

passivedns – Generate normalized records for all DNS queries and responses

```
$ passivedns -r input.pcap ⏎
  -l pdnslog.txt -L pdns_nxdomain.txt
```

- **-r** Specify pcap file to read
- **-l** Log for normal (non-error) queries
- **-L** Log for SRC error queries
- **-i** Specify interface for live DNS observation

GUI-less Packet Spelunking

tshark – Dump and analyze network traffic (aka "Wireshark in the shell")

```
$ tshark -n -r in.pcap -Y '<disp filter>'
```

-n Prevent DNS and port lookups

-r Read from pcap file instead of the network

-w Write output to a pcap file instead of the terminal

-T Output f ormat (text, fields, etc.)

-e With" -T fields", add a field to the output

-Y Protocol-aware display filter to apply

-z Statistical output modes – see man page

*See the **wireshark-filter** man page for information on building protocol-aware display filters.*

Bring Out the Big Guns

awk – Pattern scanning and processing language
Seriously powerful stuff™!

```
$ awk -F ',' '{ print $1,$6,$3 }' in.txt
```

-F Specify input field separator (default is space)

Input and output field separators can be specified in the awk script itself with the FS and OFS variables:

```
$ awk '{ FS = ","; OFS = "\t"; print $2,$4 }' in.txt
```

For More About These Fine Commands...

Use the built-in reference manual:

man Interface to the on-line reference manuals

```
$ man find
```

-k Perform keyword search through all man pages

*Use inline command help where available – many commands provide brief usage statements with the "**--help**" or "**-h**" options*

```
$ tcpdump --help
```

The Grymoire - home for UNIX wizards:

http://www.grymoire.com/Unix/

Use the Force

BASH provides many functions to improve your accuracy, speed, and efficiency – know and use them!

Tab Completion
Hit the <TAB> key to expand the first few characters of a command, directory name, filename, or variable name. If there is more than one possible option, it will complete as far as possible. Press <TAB> again to see the possible completion options.

Standard Variables

~ An alias for the current user's home directory (also available as $HOME)

$PATH The command search path

$? The exit value of the previous command

$PWD The current working directory

Command History
Cycle through previous commands by pressing the up and down arrows. Use the *history* command to see a list of the command history buffer. (BASH writes this buffer is **~/.bash_history** upon exiting, overwriting any existing contents.) Press *Ctrl-R* to search through history for commands that match a search string.

Searching: Where For Art Thou?

grep – Print lines matching a pattern

```
$ grep pattern input.txt
```

-i Case-insensitive pattern matching

-v Print lines that do not match

-c Count matching lines instead of printing them

-l Print filenames containing matching lines

-h Do not include filenames when searching multiple input files (e.g., **output*.txt**)

Working on the Chain Gang

Linux prefers small, single-purpose functions and utilities. Chain them together with the "pipe", which sends the output of one command into the next as input.

```
$ grep pattern input.txt | sort | uniq -c
```

Iteratively build a series of commands to create output that definitively addresses your requirements.

Order in the Court

sort – Sort lines alphabetically or numerically

```
$ sort input.txt
```

-n Sort numerically (5 before 10)

-r Reverse sort order

-k Specify an alternate sort field

-t Specify a field delimiter for **-k**

De-Duplication and De-Duplication

uniq – Only print consecutive matching lines once

```
$ grep pattern input.txt | uniq
```

-c Print the count of consecutive lines

Remember: Only finds consecutive matching lines! Most useful with input piped from the sort command.

```
$ grep pattern input.txt | sort | uniq
```

Redirect Output: I Don't Want to Hear You

Redirect output to a file instead of the shell itself with the "greater than" character. (Warning: It overwrites any existing contents!)

```
$ grep pattern1 input.txt > results.txt
```

Append to existing files with "double greater than"

```
$ grep pattern2 input.txt >> results.txt
```